Modern Concepts for Avionics Systems Validation Test Environments[1]

John T. B. Mayer
Leticia Montañez
James A. Roberts
Ricky D. Graves
Jet Propulsion Laboratory
California Institute of Technology
4800 Oak Grove Drive
Pasadena, CA 91109
818-354-4397
john.t.mayer@jpl.nasa.gov

Abstract— The Cassini Project is the last of the large interplanetary spacecraft missions. The Integration Test Laboratory (ITL) at the Jet Propulsion Laboratory (JPL) is part of the Cassini Project. This laboratory includes the Attitude Articulation Control Subsystem (AACS) testbed. This environment was used to not only verify and validate the AACS but was also used to develop advanced concepts and ideas for use in future testbed environments.

The concepts developed for the AACS testbed included the use of multiple computers, real-time data collection and real-time display, internet access and the incorporation of commercially available components. These concepts resulted in improved simulation fidelity, repeatable operation and reduced future operational costs. This paper will focus on a description of the testbed, lessons learned, the innovations used and future directions that will substantially reduce future testbed costs.

TABLE OF CONTENTS

1. INTRODUCTION

The Cassini Attitude and Articulation Control System (AACS) Support Equipment (SE) software was developed to exercise and test the AACS flight equipment, flight software and procedures. This project represents the continued evolution of testbed technology that started as pure analog systems, progressing to Space Shuttle's custom built testbeds [1,2], the introduction of commercial systems into the Galileo testbeds, cumulating in a hybrid custom/commercial system testbed for Cassini.

This paper will present a description of the simulation system and the innovations that were incorporated. An overview of the SE hardware will be presented but the focus will be on the simulation software. This paper will conclude with a discussion on the future innovations that are developing for this area.

2. DESCRIPTION OF THE CASSINI AACS SE

This section will present a hardware overview that supports the simulation software system [3].

AACS SE Hardware Description

The Cassini Support Equipment Software Simulation runs on a multi-processor environment. This type of environment allows massive amounts of computations to be performed at greater speeds. Communications between each processor is performed by way of reflective memory boards (VMIVME-5550). In order to keep the integrity and time homogeneity of data in a multi-processor environment, a Blackboard System [4] was employed.

The SE Software originally ran on thirteen processors. These processors included five Motorola 68030 and two SkyBolt processors for the real-time processing, one Sun Sparc 2 and five IPC's for the user interface and host computers. Later we added a HP 725 for archive purposes, a Silicon Graphics Inc. (SGI) Crimson for graphics capabilities and a generic PC for collecting bus data.

As these products neared the end of their lifecycle they were replaced with newer technology. We currently use four Motorola PowerPC's (PPC 604) for real-time processing, one Sun UltraSparc II for archive, one Sparc 20 for host functions and two generic PC's for graphics and bus

[1] U.S. Government work not protected by U.S. copyright

monitoring functions. As a byproduct of the processor replacements we were able to reduce the number of components and the complexity of the testbed. A prime example of this is how we were able to go from custom graphics on the SGI machine and 22K lines of C code, to a VRML model on a PC using about 500 lines of Java code.

processing time consumption. The portion of the software that was developed to implement this concept is named the Blackboard.

The Blackboard can be viewed as a database that is shared among processors in the SE lab. Since each processor is physically located in a different chassis in the SE lab an

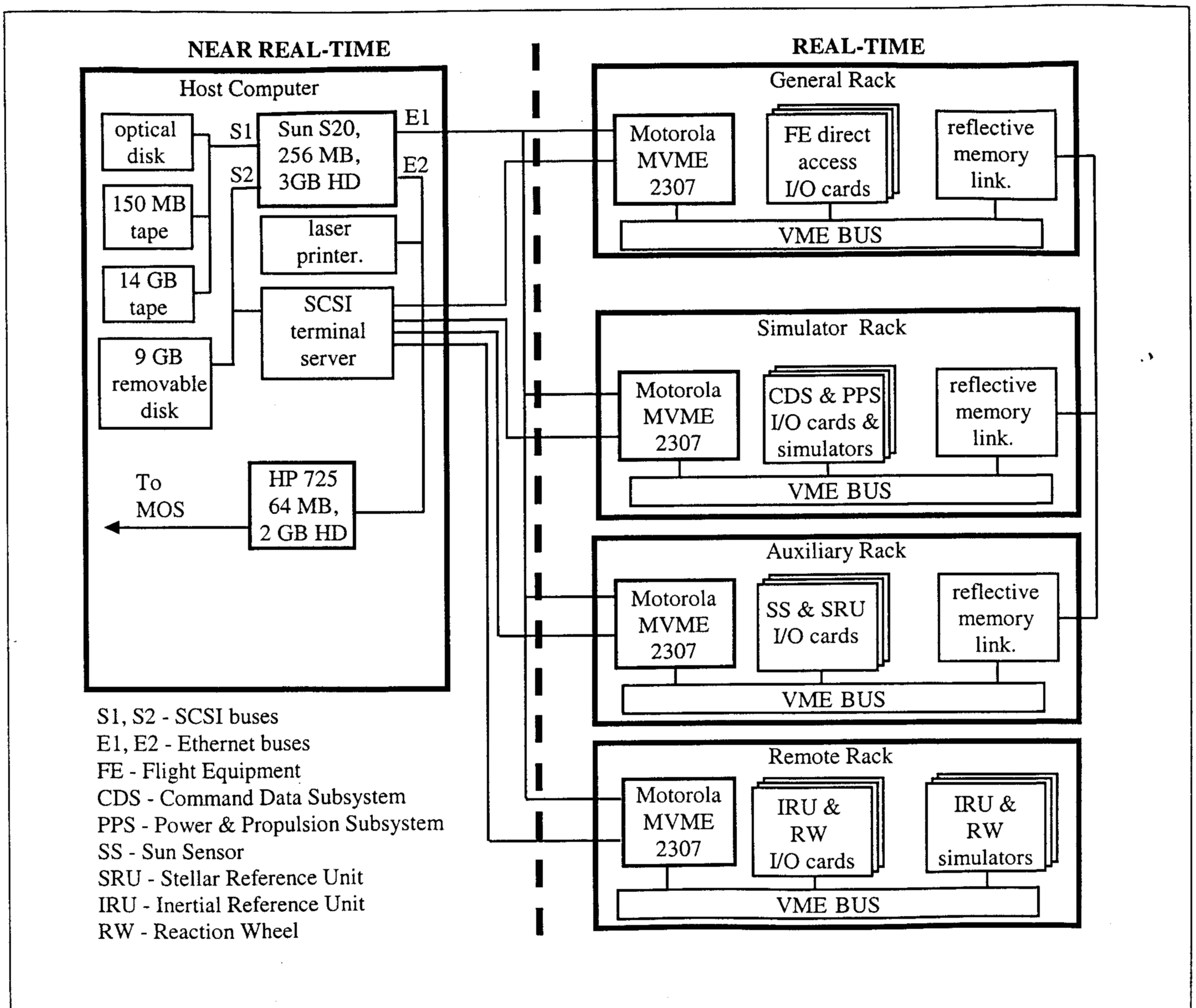

Figure 1: Cassini AACS SE Hardware Configuration

Figure 1 shows the hardware configuration presently in use.

AACS SE Software Description

The design of the SE Software is centered about the idea of responding to change [5]. This idea is based on the fact that for every change that is detected by the SE Software there is a specific response. The advantage to this design is that it leads to a low CPU processing time utilization. A global polling environment was ruled out because of its high

identical database must exist in each processor's memory. If any processor changes an entry in the database, the entry must be updated in every other database that contains the entry. This will insure that data integrity is keep throughout the system. Time homogeneity can be also kept using this same process.

Reflective memory boards are used to implement the Blackboard concept in the SE Software design. A reflective memory board exists in each chassis: General Chassis;

Auxiliary Chassis; and Simulator Chassis. Each reflective memory board broadcasts data to the other reflective memory boards when a change has been detected. This process allows each board's memory to consist of a mirror image of the other boards.

SE Software Description

The SE Software is partitioned into Non-Real-Time and Real-Time. It runs in both modes simultaneously to yield enough processing margin for the actual spacecraft simulation to run in Real-Time. Figure 2 is an overview of the SE Software broken down to a task level.

Near-Real-Time Software

The Near-Real-Time portion of the SE Software provides the simulation interface for the User. The Near-Real-Time can initialize, change and communicate to the Real-Time software via Ethernet connections. The Near-Real-Time runs under the UNIX Operating System and is composed of the Telemetry Handler, Archive, Command Parser, Display, Post Processing, History and Hardcopy data server functions. The main operations of these tasks are to control

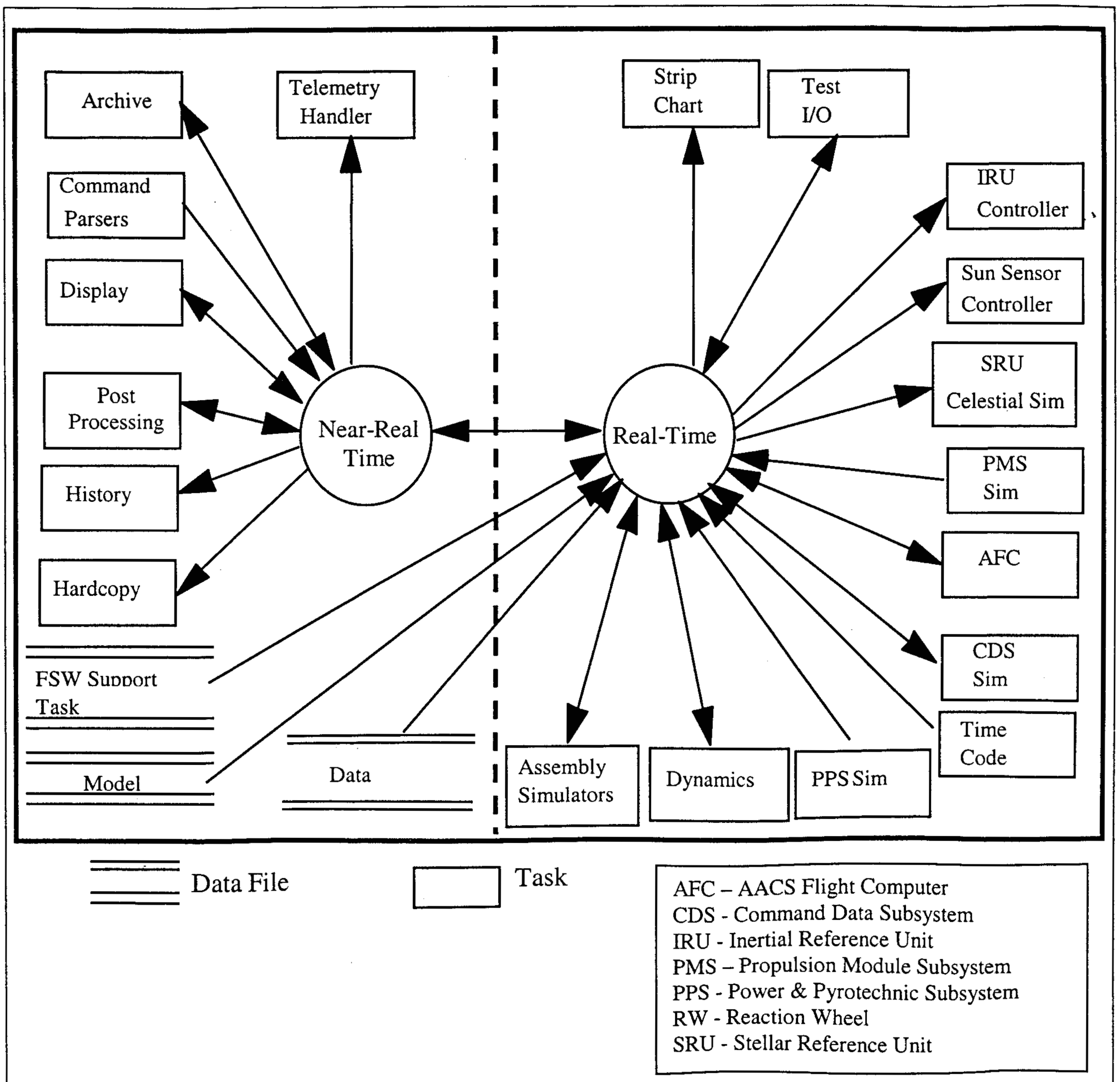

Figure 2: Cassini AACS SE Software Organization

storage, display and printing of data. The Near-Real-Time also houses initial loads and data files for the Real-Time software.

Data transfers in the Near-Real-Time are based on the UNIX pipeline concept. A pipeline is a collection of pipes between pairs of processes. A pipe is an open file connecting two processes. Data that is written into a pipe at one end may be read from the pipe at the other end. The UNIX system automatically handles synchronization, scheduling and buffering of pipes.

Real-Time Software

The Real-Time software performs the core simulation of the Cassini Spacecraft and environment. Timing is very crucial. The program execution and data transfers have to be performed within microseconds of its required time.

Although the Blackboard synchronization is performed via hardware, the Blackboard database engine is run in real-time. The Blackboard database is composed of over 2000 records. Each record is composed of many attributes that identify the data characteristics, how the record is updated, what functions are performed when the data is modified, and links to other records.

Data transfer from the Real-Time to the Near-Real-Time is performed when the data transfer flag for any Blackboard record is set. When this flag is set the data client retrieves the data for that particular Blackboard record and passes it through a datagram socket. The Near-Real-Time program receives the data. The data is then piped in a distributed method throughout the Near-Real-Time section.

Operating System

Since the SE Software runs in a multi-processor environment, an operating system exists for each computer. The SE Software consists of three different operating systems: MSDOS™, UNIX and VXWorks™. In the original system an additional operating system, SKY™, was used for the dynamics and celestial simulation processors. PowerPC processors replaced these processors.

Each SUN computer in the testbed houses and runs its own UNIX operating system. In using the UNIX operating system, networking by way of TCP/IP allows information and resources to be shared between each Sun Workstation. The only drawback of the UNIX operating system is that it does not run fast enough to support real-time testing. The operating systems that run in real-time in the SE simulation are VXWorks™.

Each PowerPC processor runs with its own copy of the VXWorks™ operating system. These processors run most of the SE Software which interface with the hardware in the testbed.

MSDOS is used in the Bus Analyzer Tool (BAT) and graphics computer. The BAT is dedicated to monitoring and capturing the 1553 bus traffic for post-test analysis. This data is sent to the Archive system via a File Transfer Protocol (FTP) connection. The graphics computer displays a Real-Time animation of the spacecraft and its environment.

Real-Time Dynamics Simulation

The Dynamics Algorithms for Real-Time Simulation (DARTS) simulator for the Cassini Spacecraft consists of a central flexible body with a number of articulated rigid-body appendages. The demanding performance requirements from the spacecraft control system require the use of a high fidelity simulator for control system design and testing. The DARTS algorithm provides a new algorithmic and hardware approach to the solution of this "hardware-in-the-loop" simulation problem. It is based upon the efficient spatial algebra dynamics for flexible multibody systems. A parallel and vectorized version of this algorithm is implemented on a low-cost microprocessor computer [6,7].

Analysis Tools

The Real-Time analysis tools used for immediate insight into testbed operation included various custom plotting tools, flight computer memory read-out decoding, pixel star scanner display and spacecraft display. Simple commercially available utilities were also used to quickly review data. These tools include such things as GREP and Netscape® Navigator.

Post-test evaluation analysis tools are used to extract, evaluate and display data. These tools are commercially available or freeware tools. These tools range from major analysis tools such as MathLab, to common business programs such as Microsoft® Excel.

3. Innovations Incorporated

The major innovations that were incorporated into the Cassini AACS SE included centering the system around a database concept, use of the internet and using an effective archive system for dissemination of test data and test results.

Database Concept

The test bed is centered on a database processing system. Inputs, such as equipment inputs, hardware interfaces, environmental parameters and user commands, are kept in a relational database with a time oriented archiving system. Outputs are extracted from the database and sent to flight equipment and the user. The various simulations, while performing complicated calculations, in general simply extract data base items and input values into the database. User commands and display items are manipulated the same way. The design and internal workings of the data base engine coordinate the inputs and outputs. Program dispatching is based on changed values or exception conditions. Coupled with a Real-Time clock, which is used to activate an update cycle, we have developed a fully functional simulation system utilizing using significant amounts of Commercial Off The Shelf (COTS) software and significantly smaller amounts of custom built software.

The data base system also coordinates activities between multiple computers demonstrating that the various processes are truly independent. The various computers communicate via reflective memory, serial I/O and local networks.

Use of the Internet

We used the Internet extensively in our data archive system. The major data used in our test analysis are: 1) the telemetry stream; 2) commands issued; 3) 1553-bus traffic and 4) environmental data. All of this data is time-stamped and collected for a complete run. It is then compressed and stored on our archive system. The data is then made available for the test analysts. All of the data is also available in Real-Time but most issues and final test acceptance requires a final analysis that cannot be performed in Real-Time. Once the data has been placed on the archive system, the analysts access it via the Internet. The analysts "log on" to the archive web site and by following a simple Web page interface, they are able to extract the data that is relevant to the issue or test that they are interested in. This data is usually downloaded to their respective workstations, which include Sun Workstations, Apple Macintoshes and PC's of all vintages. They then use their desired local tools to examine the test data. These tools range from high caliber tools such as LabVIEW™ to common tools such as Microsoft® Excel and simple text editors.

Data Archive System

We made making test data available to developers and analysts one of our primary goals early in development. That this occurred concurrently with the rise of the web is no coincidence. The web allowed users of Mac's, PC's and Unix boxes near equal access to the data, problem reports and test reports.

It is possible to use our archive system's Website based tools to look at the data. However, most analysts prefer to use their own tools with which they are more accustomed to using and are more specific to their respective tasks at hand. This is a clear indication that a basic set of tools to gain primary access the data can be defined for a complete project but the analysts should select or build their own analysis tools to solve their specific issues.

The reason for this is that at the beginning of a project, it is impossible to define all the requirements needed for what data and how the data should be analyzed. There are just too many variables to make this possible. By relying on the individual analysts to select their own tools, the analysts are more efficient and produce better results. This also eliminates the bottleneck that a centralized analysis system would create should upgrades and enhancements be needed to resolve issues or problems.

Table 1 shows the number of archive data accesses made by the project personnel for the year 1998. Note that this is post launch at a time when we reduced the number of supported testbeds from 3 to 2. This data works out to about 100 accesses/workday.

Table 1: Archive System Data Accesses

Period	Data Accesses
Jan 1998	2101
Feb 1998	2332
Mar 1998	2584
Apr 1998	2279
May 1998	2104
Jun 1998	1936
Jul 1998	2227
Aug 1998	1255
Sep 1998	1337
Nov 1998	1489
Nov 1998	1915
Dec 1998	1680

Table 2 shows the amount of data that has been archived during the year 1998. The intervals represent thirty-day periods. The simulation and telemetry numbers are uncompressed data bytes; the 1553-bus data numbers represent bytes of compressed data with an approximate 20 to 1 compression ratio.

Table 2: AACS Archive Data (bytes)

30-Day Period	Telemetry and Environment	1553-Bus Data (compressed)
1	132418332	142723432
2	193355152	244533122
3	317075199	286511679
4	249437125	317087748
5	181165428	129923737
6	177305306	315628612
7	126411562	274394502
8	206154330	402114089
9	119562744	251911716
10	375162167	787427171
11	159840857	290101035

4. Lessons Learned

The following advice is highly recommended for future testbed projects. Some of these lessons were learned from previous projects and we were glad to have taken the advice.

Get Involved Early In Requirements Definition

Software and hardware engineers need to be involved in each other's requirements definition from concept through delivery. The testbed engineers should also be involved with the spacecraft or instrument's requirements. This lesson goes to the heart of developing an economical project. By following this rule, we were able to avoid costly redesigns.

Adhere To Strict Configuration Control

We highly recommend following a rigid configuration control policy. In most cases it is well worth the expense of adapting or developing the tools necessary to implement the policies. With the proper tool, software developers will follow the configuration controls. In those few cases were we failed to follow this rule, code was lost or errors were introduced resulting in added expense.

Continuity of Personnel

We recommend that a core group of developers be maintained over the life of a project. We seriously suffered from continual turnover of our dynamics domain experts costing an estimated additional four man-years of additional work.

Build and Release Often

Plan for the capability to incorporate changes and make quick deliveries. Plan on making partial or stopgap deliveries. The Cassini AACS SE software is an evolving entity. The fidelity of the models and simulators are continually being tightened. Capabilities based on user feedback need quick turnaround to provide needed test functions. It is desirable to be able to make updates where only one change at a time is delivered. Users tend to lack confidence in a system where several changes have been made in a delivery.

Deliver the Development Tools with the System

A software delivery should also include the tools needed to create the software delivery. Tools evolve at a rapid rate. Often times there is a need to run an old version of a software delivery for problem resolution purposes. This capability can be lost if the toolset originally used has advanced to where it is no longer backward compatible.

5. FUTURE DIRECTIONS AND INNOVATIONS

The Cassini AACS SE has demonstrated that new innovations can significantly reduce development costs. Areas that we are currently considering for future systems include an enhanced use of the internet, an integrated upgrade and configuration process, existing COTS software and ultra-high level system development languages.

Enhanced Use of the Internet

Future evolution of Real-Time simulation systems will include the use of the Internet. The Internet can be used for inter-process and inter-computer communication. With our current system we have demonstrated that a loosely coupled set of computers and tasks can be combined into one effective simulation system. With the introduction of the Internet, it should be possible to organize a set of computers or web sites to perform the same functions. The obvious questions that would have to be investigated include the various time lags involved, security concerns, reliability and reconfiguration. These issues are not insurmountable. The possible use of dedicated networks with gateways could provide the required security and available transmission bandwidth to make this a reality.

Component Upgrade and Configuration Management

Upgrading of individual components could become transparent to the system thus reducing the required management and system coordination costs. This could be achieved with planned backward compatibility. For example, the individual components in a system would identify its "version level" and the "version level" of any required external components or interfaces. So long as that one maintains backward compatibility, another component could be upgraded, thus supplying its new version (n) and its previous versions (n-1, n-2 ... 1). As a result the new version would be transparent to its existing interfaces. It is also reasonable to think about a component interacting with the rest of the system using multiple previous versions. This will greatly simplify the upgrade and maintenance issues and costs that now dominate the industry.

COTS Software

The use of COTS components will probably increase in the future. Our current system clearly demonstrates that COTS components can be incorporated into high fidelity systems. The benefits would include the use of low cost, sometimes free, development systems, an availability of technical personnel, reduced maintenance costs, non-obsolesce, and the ability to execute projects in a geographically dispersed area. The main benefit would be the ability to incorporate COTS development systems and tools such as JAVA™, VxWorks, LINUX and Windows (CE or NT).

There already exists a large array of software that although was developed with commercial business applications in mind, can be utilized in a simulation system. It is even conceivable that something like a spreadsheet application could be effectively used. The major value of using COTS is lower initial and long-term costs, larger available workforce to support the system and commonality with other systems. The obvious disadvantages involve having less direct control over the product. This latter issue is becoming less serious. It is true that if COTS components are used and one runs into a bug, one does not have the control to dictate an immediate fix. But this is mitigated by two factors: 1) with larger numbers of people using a COTS product, bugs get identified and fixed earlier in a products life cycle; and 2) usually there are multiple ways of doing something. So if one approach has a bug there is usually a less efficient method that can be employed to get around the problem. Also, in a competitive environment, there may also be other vendors with competing products that can be employed to resolve the issue.

Existing Types of COTS Software

What types of COTS could be used? Data base engines, user interfaces, interpreters and compilers (JAVA™, Basic, C++, Visual Basic), development environments, support tools, report generators, spreadsheets, or general device drivers.

Use of PC's and the Windows™ system may also be possible. We are currently looking at Windows NT™ as a possible alternative. It seems that the answer is "yes" but the jury is still out. Certainly, the differences between Windows NT™ and UNIX are obvious and the political concerns are also apparent, but the economic concerns favor a widely used, inexpensive operating system.

Potential Future COTS Software

It appears that we are following a path where the major components of a simulation system may all be COTS or inherited software. The main items left to the specific projects would consist of device drivers and high level algorithms. The latter could be developed using high level visual tools to describe the system. Then automatic source code generators could be used to create the actual software. Going one step further, it may even be possible to define the flight system in an ultra-high level language and then automatically generate not only the "flight code" used by the actual system, but also the simulation system software needed to exercise the flight software.

6. Summary and Conclusions

The issues and advancements presented will significantly reduce the overall costs of future support equipment simulation systems. Couple the potential economic savings with the prospect of inheriting or bequeathing to future projects this simulation infrastructure, the much touted "faster, better, cheaper" mantra may actually be possible.

References

[1] John T. B. Mayer, *The Build System – Integration and Management of Large Software Avionics Systems,* presented at the Space Enhancing Technological Leadership, American Astronautical Society, 1980 Annual Meeting, October 20-23, 1980.

[2] John T. B. Mayer, *The Space Shuttle Vehicle Checkout Involving Flight Avionics Software,* presented at the AIAA Computers in Aerospace III Conference, San Diego, CA, October 26-28, 1981.

[3] Ricky Graves, *AACS Support Equipment Hardware – Design Requirements and Description Document, Rev. C,* (Internal Document), Jet Propulsion Laboratory, Pasadena, CA, 1994.

[4] Robert Engelmore and Tony Morgan, *Blackboard Systems (The Insight Series in Artificial Intelligence),* Reading, MA, Addison-Wesley, 1988.

[5] Leticia Montañez, *AACS Cassini Support Equipment Software – Design Requirements and Description Document,* Interoffice Memorandum 3413-96-122 CAS, (Internal Document), Jet Propulsion Laboratory, Pasadena, CA, 1996.

[6] A. Jain and G. K. Man, "Real-Time Dynamics Simulation of the Cassini Spacecraft Using DARTS. Part I. Functional Capabilities and the Spatial Algebra Algorithm," *Proceedings of the Fifth NASA/NSF/DOD Workshop on Aerospace Computational Control,* Jet Propulsion Laboratory, Pasadena, CA, 1993.

[7] A. Fijany, J. A. Roberts, A. Jain and G. K. Man, "Real-Time Dynamics Simulation of the Cassini Spacecraft Using DARTS. Part II. Parallel/Vectorized Real-Time Implementation," *Proceedings of the Fifth NASA/NSF/DOD Workshop on Aerospace Computational Control,* Jet Propulsion Laboratory, Pasadena, CA, 1993.

Biography

John Mayer is a Senior Engineer in the software development and system integration and test fields. He has developed flight and ground software at the Jet Propulsion Laboratory and Intermetrics Inc. since the mid-1970's. He is presently supporting the SE upgrade development for the Cassini Program. He has been involved with the space shuttle, space station, Galileo, Cassini and GPS. He has a BSEE and MSEE from Stanford University.

Leticia Montañez *is a Senior Engineer in the system integration and test field. She has been instrumental to the AACS system testing for the Galileo and Cassini projects at the Jet Propulsion Laboratory since the mid-1980's. She is presently responsible for the Integration and Test Lab for the flight phase of the Cassini Program. She has a BSCS from California State University, Los Angeles.*

James Roberts *is a Senior Engineer in the Real-time software development field. He has developed flight and ground software at the Jet Propulsion Laboratory and Rockwell International since the late-1970's. He recently served as the lead Software engineer for the SE development for the Cassini Program. He has been involved with the space shuttle, Galileo and Cassini programs. He attended California State Polytechnic University, San Luis Obispo.*

Ricky Graves *is a Senior Hardware Engineer in the system integration and test fields. He has developed the hardware for integration and support test systems since 1985. He recently served as a SE Hardware Cognizant engineer in support of the Cassini Program at the Jet Propulsion Laboratory. He has been involved with the Galileo and Cassini projects. He has a BSET from California State Polytechnic University, Pamona.*